




Enhancing Virtual and Augmented Reality Interactions with a MediaPipe-Based Hand Gesture Recognition User Interface



Beom Jun Jo¹ , Seok-Kyoo Kim¹ , SeongKi Kim^{2*} 

¹ Department of Game Design and Development, Sangmyung University, Seoul 03016, Korea

² National Centre of Excellence in Software, Sangmyung University, Seoul 03016, Korea

Corresponding Author Email: skkim9226@gmail.com

<https://doi.org/10.18280/isi.280311>

ABSTRACT

Received: 31 January 2023

Accepted: 18 May 2023

Keywords:

virtual reality, augmented reality, gesture classification, MediaPipe, user interface

Interaction with virtual objects is crucial for presence in Virtual Reality (VR) and Augmented Reality (AR) applications. However, controllers are still predominantly used for operations in virtual spaces. Hand gestures offer a more intuitive approach than keyboards and mice for interactions in these environments. In previous research, hand motion classification was implemented using only simple heuristics. This study introduces a User Interface (UI) employing MediaPipe and artificial intelligence to utilize hand gestures as an input device. Unlike the previous research, which could only identify one gesture, the current implementation successfully classifies three gestures with 95.4% accuracy: pointer, pick, and fist. Efforts were made to optimize the process, including the examination of multi-threading and PyWin32. While multi-threading did not yield significant improvements, the use of PyWin32 resulted in approximately three times higher Frames Per Second (FPS) compared to the absence of PyWin32. Further gestures can potentially be added to enhance the system's capabilities. This line of research has potential applications in diverse fields such as gaming, simulation, rehabilitation, and smart home technology.

1. INTRODUCTION

Virtual Reality (VR) is an artificial environment rendered by a computer, while Augmented Reality (AR) superimposes virtual objects over physical ones. With the emergence of these technologies, users are required to interact with virtual objects. However, early implementations offered limited interaction methods, allowing users to merely observe objects without engaging with them. As technology advanced, various equipment for interaction between virtual and physical objects have been developed. For example, VR treadmills [1] can simulate walking in virtual spaces, and haptic suits [2] can deliver sensations from the virtual environment. Nevertheless, controllers remain a common choice due to cost and space constraints, reducing presence and immersion in VR and AR experiences.

Meta Quest 2 has adopted hand tracking technology [3]. When the hand tracking option is enabled, users can interact without controllers, utilizing three gestures [4]: "point and pinch," "pinch and scroll," and "palm pinch." As these gestures are limited in number, they are suitable only for simple games or activities.

Just Dance serves as an example [5, 6]. This game has been released on various platforms, with playing methods differing across devices. While controllers such as smartphones or Nintendo Switch Joy-Cons are typically used, body gestures were employed on the Kinect and Xbox platforms, providing users with a more natural interaction method. Interacting through body gestures in games or VR/AR content frees users from cost and space issues and can prevent accidents caused

by detached controllers, which are usually mitigated by wrist straps.

In this study, a hand gesture-based user interface is implemented as part of hand gesture classification. As hand gestures resemble body language, the user interface is more intuitive than using a mouse. This interaction method offers extensive applicability, including hands-on games, rehabilitation [7], fitness programs like Apple Fitness+ [8], and sign language [9]. Furthermore, it can be employed in automobile control and smart home applications.

Previous research [10] classified hand movements using angles between vectors, obtained through point coordinates. However, this method necessitated numerous angle constraints for accurate motion estimation. In contrast, the current study employs artificial intelligence (AI) to facilitate the addition of more gestures. Nevertheless, this approach only operates on Windows and has not yet been applied to the ultimate goal: VR devices.

The contributions of this study include: (1) proposing a basic method for classifying hand gestures, which tracks hands through MediaPipe and utilizes a pre-trained model for gesture classification; (2) implementing a user interface for controller-free operation; and (3) suggesting an optimization method for the implementation.

The remainder of this study is organized as follows: Section 2 presents related work and commercial products exemplifying hand tracking and gesture classification solutions. Section 3 details the realization of the proposed system. Section 4 concludes the study.

2. RELATED WORK

This section describes existing methods and current research for using hands as the user interface.

2.1 Existing methods

Meta released the Meta Quest 2 in 2020, which supported a hand tracking option. Three supported gestures are illustrated in Figure 1. Although there are only a few movements, they

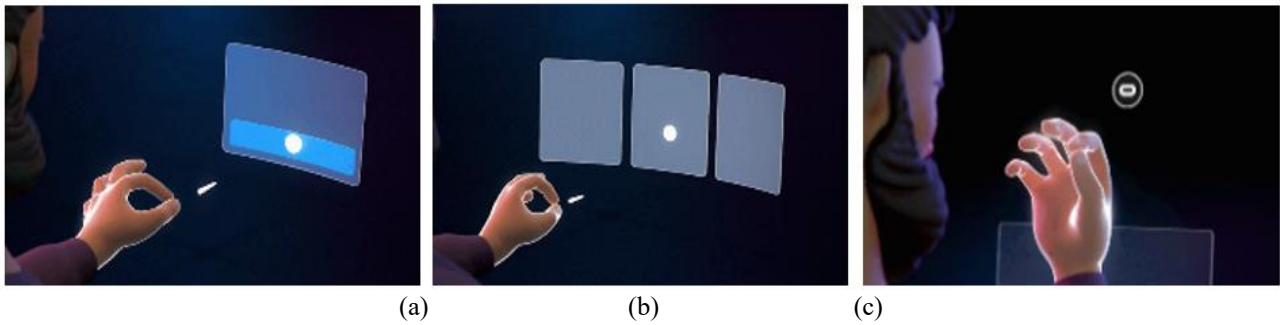


Figure 1. Three gestures supported by Meta Quest 2 [4] (a) point and pinch; (b) pinch and scroll; (c) palm pinch

2.2 Methods for hand tracking

In general, methods for tracking hands can be categorized into three approaches: attaching sensors to hands, using cameras with special functions, and utilizing machine learning with a normal camera.

2.2.1 Sensors attached directly to the hand

The first approach involves attaching sensors directly to the hands. This method has been developed since the early stages of hand tracking. As it only requires attaching hardware to the player's body, it is the easiest to implement and the most accurate method. It is actively used in motion captured films, and professional products such as MANUS metagloves [11] can be found. HTC's VIVE tracker [12] is a popular commercially available product. Additionally, SlimeVR [13] is an open-source project. In the case of such trackers, full tracking is possible by attaching them to the entire body.

In this approach, users directly attach sensors to their bodies. Generally, sensors include gyroscopes and accelerometers,

which calculate and track necessary information such as angle, acceleration, and gravity.

2.2.2 Using a specific camera

The second method utilizes cameras with special functions, such as depth sensors. This method has been actively studied, and commercially available products use specific sensors. Examples include Kinect [14] and Leap Motion [15, 16], which employ a depth camera and an infrared (IR) camera, respectively. Microsoft's Azure Kinect implements the Amplitude Modulated Continuous Wave Time-of-Flight (AMCW ToF) principle and can measure up to millimeters [17, 18].

Using Azure Kinect as an example, the device employs both an RGB camera and a depth camera for recording. Kinect's depth cameras can measure Z coordinates through depth measurements and IR images. Thus, when combined with pixel information from an RGB camera, the depth of a specific pixel can be obtained, allowing for accurate depth tracking (See Figure 2).

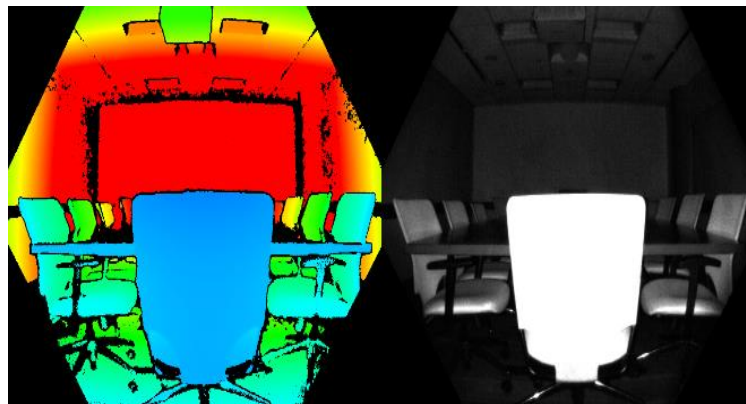


Figure 2. (Left) Depth map obtained by Azure Kinect (Right) Image obtained by IR camera [14]

2.2.3 Using artificial intelligence

The third method involves using machine learning with a regular camera. The most significant advantage of this method

is its compatibility with standard cameras. Mueller et al. [19] demonstrate 3D hand pose tracking via GeoConGAN [20] and RegNet [21]. Another hand tracking solution is MediaPipe

[22], which utilizes machine learning and is an open-source platform that supports various platforms. Recent studies aim to create a virtual hand based on information beyond tracking. Grishchenko et al. [23] employ an improved model that combines existing BlazePose and MediaPipe to track not only the full body but also detailed hand movements, enabling the creation of a virtual body using GHUM [24].

The third method has cost advantages since it operates with a standard camera. Additionally, this method can be used if screen input can be received, regardless of camera performance.

2.3 Classification

Gesture classification allows for accurate motion determination. A previous study [10] examined how to use hand gestures in mobile games. A simple heuristic was developed to recognize the hand shape as a gun. Several attempts were made at coordinates and angles to recognize the exact gun shape. However, using artificial intelligence for this purpose could facilitate adding new movements.

Hand classification is typically conducted with images alone, without tracking. Even when action classification is considered, it tends to rely solely on RGB data from images or videos. For example, Gadekallu et al. [25] classify using a CNN-crow search algorithm, but only with images. However, the study in question first tracks and then classifies actions based on this information. As a one-dimensional array is used for the classification process, it is faster than when using images.

3. IMPLEMENTATION AND RESULTS

In the previous study [10], an action was classified through a simple heuristic by using ManoMotion [26] on Unity. This

research aims to devise a method that could use hand gestures as a user interface. This section describes the implementation and results of the proposed approach, as well as performance optimization. To implement the method, Ryzen 4800h and GTX 1660Ti hardware were used, along with Python, Google’s MediaPipe, and Windows for hand tracking software.

3.1 Gesture classification model

This subsection discusses hand tracking prior to the gesture classification of hands, as hand tracking is utilized. While it is feasible to classify solely based on images without extracting information about the hands, the input image size must be consistent for the model. Thus, it is necessary to make the image identical only by cropping the hand. However, the method presented in this paper does not necessitate a process for adjusting the image size.

After processing with MediaPipe, hand coordinates are obtained. Figure 3 displays the relevant landmarks. Using the landmarks in Figure 3, gestures shown in Figure 4 were recognized. Figure 4 depicts the hand gestures utilized for controlling a cursor in this research. In Figure 4, the pointer gesture moves the mouse cursor. The green dot in Figure 4(a) corresponds to point 8 in the landmarks shown in Figure 3. The cursor moves along this point. The pick gesture in Figure 4(b) performs a drag, while the clenched gesture in Figure 4(c) executes a click.

Figure 5 presents the architecture of the classification model. MediaPipe generates coordinates for each landmark as an output. The classification model receives a one-dimensional array containing coordinates obtained through MediaPipe as input [27]. The training process is similar. Ultimately, the model classifies four gestures in Figure 4: Open hand, fist, pointer, and pick. Table 1 lists the parameters used by the model.

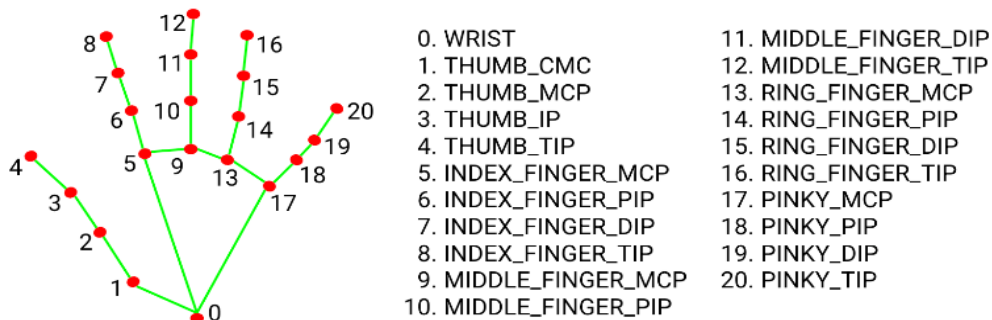


Figure 3. Hand landmarks tracked by MediaPipe

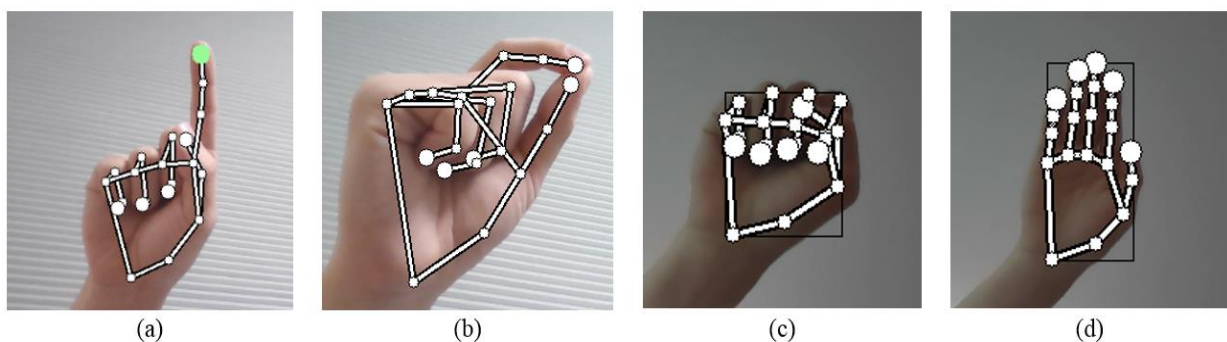


Figure 4. Images of gestures for controlling a cursor (a) a pointer gesture to move a cursor; (b) a pick gesture to drag; (c) a fist gesture to click; (d) an open hand gesture

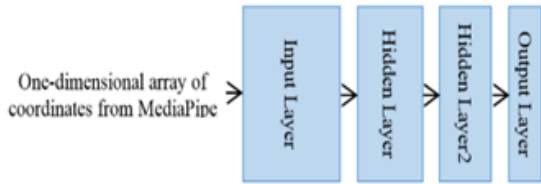


Figure 5. Suggested architecture of the classification model

Table 1. The parameters of the model

Parameter	Description
Optimizer	Adam optimizer
Loss function	sparse_categorical_crossentropy
Number of epochs	1000
Batch size	128
Number of dropout	2
Number of dense layer	3
Activation function at intermediate layer	Relu
Activation function at output layer	Softmax

3.2 Implementation and results

The method outlined in Subsection 3.1 was implemented on Windows using Python and MediaPipe. The following steps were executed:

1. Capture an image from the webcam.
2. Pass the image through the MediaPipe classifier. Various information about the hand is obtained after processing.
3. The AI-based classifier model identifies the current gesture among predetermined hand gestures.
4. Control the mouse according to the results from Step 2.
5. Repeat Steps 1-3.

Coordinates of hand landmarks are acquired through Steps 1 and 2. These coordinates were used to capture and store hand gestures for future use. A classifier is created when the model is trained with collected coordinates by capturing dozens of samples.

Subsequently, the landmark coordinates are input into the AI-based classifier in Step 3, and the corresponding gesture is classified. Each gesture is assigned functions to perform in Step 4.

Figure 6 demonstrates the implementation results. In Figure 6(a), the pointer gesture allows the cursor to move across the desktop, following the green dot on the fingertip. In Figure 6(b), the pick gesture successfully performs dragging, as if holding and lifting. The fist gesture in Figure 6(c) executes a click. The open hand gesture in Figure 6(d) can be utilized as a resting action for the hand.

Figure 7 displays the confusion matrix of the proposed model. The vertical axis represents actual values, and the horizontal axis represents predicted values. The numbers indicate the open hand in Figure 3(d), fist in Figure 3(c), pointer in Figure 3(a), and pick gesture in Figure 3(b) in order from zero. When row numbers and column numbers are the same in the matrix, it indicates the number of correct predictions made by the model. By dividing the sum of these values by the total, the accuracy can be calculated. As a result, the proposed model achieves 95.4% accuracy $((403+326+339+86)/1210 \times 100)$.

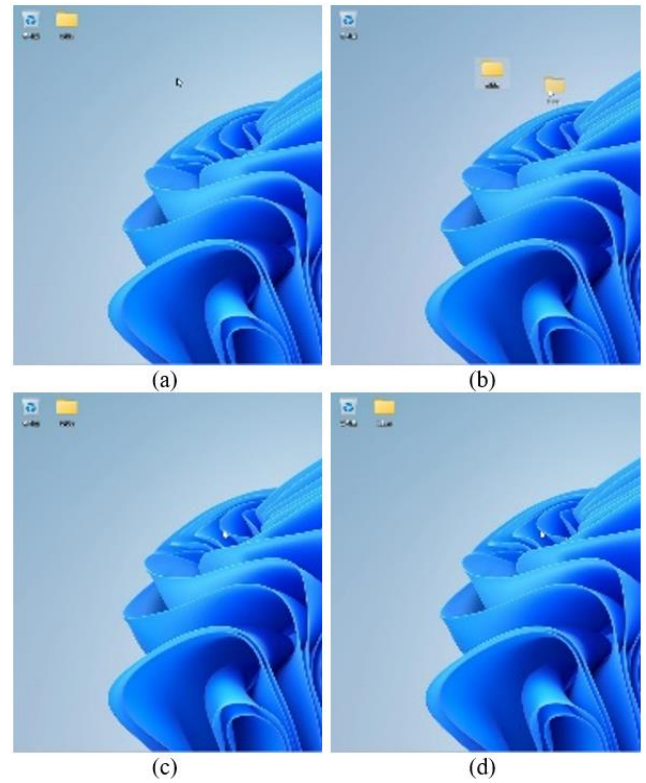


Figure 6. Running implementation (a) a pointer gesture; (b) a pick gesture performing drag; (c) a fist gesture executing a click; (d) an open hand gesture (Details can be watched at the <https://www.youtube.com/watch?v=OjzplypSBpA>)

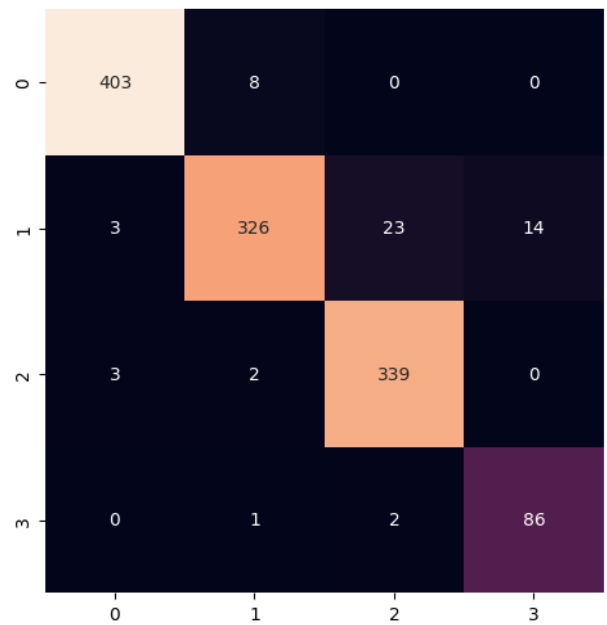


Figure 7. Confusion matrix of suggested model

3.3 Performance optimization

In Sub-sections 3.1 and 3.2, the AI-based method for using hand gestures and provided examples of developed gestures were described. However, the method could be challenging to use if its performance is significantly decreased. To address this issue, research on performance improvement was conducted, which is detailed in this subsection.

3.3.1 Multi-threading

To optimize performance, multi-threading was employed. As most recent CPUs support multi-threading, this approach has been adopted. Table 1 presents a comparison of FPS when multi-threading is used and when it is not used.

Table 2 compares the average framerate with and without multi-threading for one minute. The first row compares the framerate when nothing is drawn on the camera, and the second row compares the framerate when a hand is displayed on the camera. In this case, there was no significant difference in the average framerate; however, the framerate was more stable when multi-threading was used.

Table 2. Average performance between with and without multi-threading for 1 minute (Unit: FPS)

Case	No multi-threading	Multi-threading
Nothing on Camera	27.8	27.68
Hand on Camera	17.96	18.13

3.3.2 Graphic User Interface (GUI) library

To further optimize performance, PyWin32 instead of PyAutoGUI was used. There are PyAutoGUI [28] and PyWin32 [29] libraries available for controlling the GUI. PyAutoGUI is a cross-platform library that enables mouse and keyboard control for automating interactions. This library provides functions for moving the cursor and clicking, sending keystrokes, finding a recognized image on screen, and locating an application's window. Beyond simple keyboard and mouse usage, complex tasks can be accomplished by implementing macros using these functions.

PyWin32 is a library that allows utilizing Win32 API [30] functions in Python. Win32 API is a C language-based set of functions supporting UI control, Windows console administration, storage data access, graphics, network, and security in Windows. However, as the name suggests, it can only be used in Windows.

Table 3 compares the average framerate according to the libraries used to move the cursor for one minute. Since both libraries operate only when hands are recognized, there is no speed difference, as shown in the first row of Table 3. When using PyWin32, the average performance was about three times higher than when using PyAutoGUI. Real-time usage was challenging when using PyAutoGUI.

Table 3. Average performance between PyAutoGUI and PyWin32 for 1 minute (Unit: FPS)

Case	PyAutoGUI	PyWin32
Nothing on Camera	28.22	27.66
Hand on Camera	6.32	17.97

4. CONCLUSIONS

In this paper, developing a hand gesture-based interface using a regular webcam and AI was discussed, as this can enhance user immersion in the VR/AR field, serve as assistive control, or simplify human-computer interaction. Additionally, our hand gesture-based interface can be applied to games, rehabilitation, and sign language. Apart from the aforementioned advantages, the proposed method is cost-effective because it only requires a widely available camera. This research successfully recognized three gestures to control cursor movement with 95.4% accuracy. During

implementation, performance optimization was necessary, so suitable libraries were investigated and found that PyWin32 outperformed PyAutoGUI.

However, this research has a limitation in that the number of recognizable gestures is limited. In the future, we plan to develop an improved model to increase the number of gestures and further implement continuous gesture classification on VR devices.

ACKNOWLEDGEMENT

This research was funded by a 2022 Research Grant (2022-A000-0048) from Sangmyung University. Seok-Kyoo Kim and SeongKi Kim are the corresponding authors. All of the implementations are available at <https://github.com/torama06/hand-gesture-mouse>. To increase the understandability of all the results presented in this paper, we have created a video and uploaded it to <https://www.youtube.com/watch?v=OjzplypSBpA>.

REFERENCES

- [1] Virtuix. Omni One-The Future of Gaming. <https://omni.virtuix.com/>, accessed on Dec. 20, 2022.
- [2] bHaptics. Most Advanced Full Body Haptic Suit. <https://www.bhaptics.com/tactsuit/tactsuit-x40>, accessed on Jan. 9, 2022.
- [3] Meta. Hand Tracking Privacy Notice. <https://www.meta.com/help/quest/articles/accounts/privacy-information-and-settings/hand-tracking-privacy-notice/>, accessed on Jan. 11, 2023.
- [4] Meta. Getting Started with Hand Tracking on Meta Quest Headsets. <https://www.meta.com/help/quest/articles/headsets-and-accessories/controllers-and-hand-tracking/hand-tracking-quest-2/>, accessed on Dec. 23, 2022.
- [5] Ubisoft. Just Dance 2023 Edition. <https://www.ubisoft.com/en-us/game/just-dance/2023>, accessed on Jan. 23, 2023.
- [6] Fandom. Just Dance series. [https://justdance.fandom.com/wiki/Just_Dance_\(series\)](https://justdance.fandom.com/wiki/Just_Dance_(series)), accessed on Jan. 19, 2023.
- [7] Zhou, H., Hu, H. (2008). Human motion tracking for rehabilitation—A survey. *Biomedical Signal Processing and Control*, 3(1): 1-18. <https://doi.org/10.1016/j.bspc.2007.09.001>
- [8] Apple. Apple Fitness+. <https://www.apple.com/apple-fitness-plus/>, accessed on Jan. 20, 2023.
- [9] GitHub. Hand_Asl_Recognition. https://github.com/cortictchnology/hand_asl_recognition, accessed on Jan. 26, 2023.
- [10] Beom Jun, J.O., Seong Ki, K.I.M., Seok-Kyoo, K.I.M. (2022) A study on hand gesture classification for mobile game. *Korean Society For Computer Game*, 35(3): 1-7. <https://doi.org/10.21493/kscg.2016.29.2.1>
- [11] MANUS. MANUS-Finger & Full-body Tracking for Mocap and VR. <https://www.manus-meta.com/>, accessed on Dec. 23, 2022.
- [12] HTC Corporation. VIVE | VIVE Tracker. <https://www.vive.com/kr/accessory/vive-tracker/>, accessed on Dec. 23, 2022.
- [13] SlimeVR. SlimeVR Docs. <https://docs.slimevr.dev/>,

- accessed on Dec. 24, 2022.
- [14] Microsoft. Azure Kinect DK-Develop AI Models | Microsoft Azure. <https://azure.microsoft.com/en-us/products/kinect-dk/#overview>, accessed on Dec. 23, 2022.
- [15] Ultraleap. Digital Worlds That Feel Human-Ultraleap. <https://www.ultraleap.com/>, accessed on Dec. 23, 2022.
- [16] Ultraleap. How Hand Tracking Works. <https://www.ultraleap.com/company/news/blog/how-hand-tracking-works/>, accessed on Dec. 27, 2022.
- [17] Microsoft. Azure Kinect DK Depth Camera | Microsoft Learn. <https://learn.microsoft.com/en-us/azure/kinect-dk/depth-camera>, accessed on Dec. 24, 2022.
- [18] Lun, R., Zhao, W. (2015). A survey of applications and human motion recognition with Microsoft Kinect. *International Journal of Pattern Recognition and Artificial Intelligence*, 29(05): 1555008. <https://doi.org/10.1142/S0218001415550083>
- [19] Mueller, F., Bernard, F., Sotnychenko, O., Mehta, D., Sridhar, S., Casas, D., Theobalt, C. (2018). Generated hands for real-time 3d hand tracking from monocular rgb. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 49-59.
- [20] GitHub. bread1984/GeoConGAN: This Project Implement The 3D Hand Joint Detection Algorithm GeoConGAN. <https://github.com/breadbread1984/GeoConGAN>, accessed on Dec. 23, 2022.
- [21] Zhang, F., Bazarevsky, V., Vakunov, A., Tkachenka, A., Sung, G., Chang, C.L., Grundmann, M. (2020). MediaPipe hands: On-device real-time hand tracking. *arXiv Preprint arXiv: 2006.10214*. <https://doi.org/10.48550/arXiv.2006.10214>
- [22] Sung, G., Sokal, K., Uboweja, E., Bazarevsky, V., Baccash, J., Bazavan, E.G., Chang, C.L., Grundmann, M. (2021). On-device real-time hand gesture recognition. *arXiv Preprint arXiv: 2111.00038*. <https://doi.org/10.48550/arXiv.2111.00038>
- [23] Grishchenko, I., Bazarevsky, V., Zanfira, A., Bazavan, E.G., Zanfira, M., Yee, R., Raveendran, K., Zhdanovich, M., Grundmann, M., Sminchisescu, C. (2022). BlazePose ghum holistic: real-time 3D human landmarks and pose estimation. *arXiv Preprint arXiv: 2206.11678*. <https://doi.org/10.48550/arXiv.2206.11678>
- [24] GitHub. GHUM & GHUML. <https://github.com/google-research/google-research/tree/master/ghum>, accessed on Jan. 12, 2023.
- [25] Gadekallu, T.R., Alazab, M., Kaluri, R., Maddikunta, P.K.R., Bhattacharya, S., Lakshmana, K. (2021). Hand gesture classification using a novel CNN-crow search algorithm. *Complex & Intelligent Systems*, 7: 1855-1868. <https://doi.org/10.1007/s40747-021-00324-x>
- [26] ManoMotion. ManoMotion. <https://www.manomotion.com/>, accessed on Dec. 25, 2022.
- [27] GitHub. Hand Gesture Mouse. <https://github.com/torauma06/hand-gesture-mouse>, accessed on Jan. 12, 2023.
- [28] Sweigart, A. PyAutoGUI's Documentation. <https://pyautogui.readthedocs.io/en/latest/>, accessed on Jan. 1, 2023.
- [29] GitHub. Pywin32. <https://github.com/mhammond/pywin32>, accessed on Jan. 2, 2023.
- [30] Microsoft. Win32 Apps. <https://learn.microsoft.com/en-us/windows/win32/apiindex/windows-api-list>, accessed on Jan. 2, 2023.